Few-Shot and Zero-Shot Learning for Historical Text Normalization

Bollmann, Marcel; Korchagina, Natalia; Søgaard, Anders

Published in:
Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)

DOI:
10.18653/v1/D19-6112

Publication date:
2019

Document version
Publisher's PDF, also known as Version of record

Few-Shot and Zero-Shot Learning for Historical Text Normalization

Marcel Bollmann[♣] and **Natalia Korchagina**[◇] and **Anders Søgaard**[♣]
[♣]Department of Computer Science, University of Copenhagen
[◇]Institute of Computational Linguistics, University of Zurich
marcel@di.ku.dk, korchagina@ifi.uzh.ch, soegaard@di.ku.dk

Abstract

Historical text normalization often relies on small training datasets. Recent work has shown that multi-task learning can lead to significant improvements by exploiting synergies with related datasets, but there has been no systematic study of different multi-task learning architectures. This paper evaluates 63 multi-task learning configurations for sequence-to-sequence-based historical text normalization across ten datasets from eight languages, using autoencoding, grapheme-to-phoneme mapping, and lemmatization as auxiliary tasks. We observe consistent, significant improvements across languages when training data for the target task is limited, but minimal or no improvements when training data is abundant. We also show that zero-shot learning outperforms the simple, but relatively strong, identity baseline.

1 Introduction

Historical text normalization is the task of mapping variant spellings in historical documents—e.g., digitized medieval manuscripts—to a common form, typically their modern equivalent. The aim is to make these documents amenable to search by today's scholars, processable by NLP tools, and accessible to lay people. Many historical documents were written in the absence of standard spelling conventions, and annotated datasets are rare and small, making automatic normalization a challenging task (cf. Piotrowski, 2012; Bollmann, 2018).

In this paper, we experiment with datasets in eight different languages: English, German, Hungarian, Icelandic, Portuguese, Slovene, Spanish, and Swedish. We use a standard neural sequence-to-sequence model, which has been shown to be competitive for this task (e.g., Korchagina, 2017; Bollmann, 2018; Tang et al., 2018). Our main focus is on analyzing the usefulness of multi-task learning strategies (a) to leverage whatever supervision is available for the language in question (*few-shot learning*), or (b) to do away with the need for supervision in the target language altogether (*zero-shot learning*).

Bollmann et al. (2017) previously showed that multi-task learning with grapheme-to-phoneme conversion as an auxiliary task improves a sequence-to-sequence model for historical text normalization of German texts; Bollmann et al. (2018) showed that multi-task learning is particularly helpful in low-resource scenarios. We consider three auxiliary tasks in our experiments—grapheme-to-phoneme mapping, autoencoding, and lemmatization—and focus on extremely low-resource settings.

Our paper makes several contributions:

(a) We evaluate 63 multi-task learning configurations across ten datasets in eight languages, and with three different auxiliary tasks.

(b) We show that in few-shot learning scenarios (ca. 1,000 tokens), multi-task learning leads to robust, significant gains over a state-of-the-art, single-task baseline.[1]

(c) We are, to the best of our knowledge, the first to consider *zero-shot historical text normalization*, and we show significant improvements over the simple, but relatively strong, identity baseline.

While our focus is on the specific task of historical text normalization, we believe that our results can be of interest to anyone looking to apply multi-task learning in low-resource scenarios.

[1]We note that 1,000 tokens is more instances than is typically considered in few-shot learning; e.g., Kimura et al. (2018) use up to 200 instances. We argue that for structured prediction it is reasonable to assume more data, yet we also consider scenarios down to as little as 100 instances.

Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo), pages 104–114
Hong Kong, China, November 3, 2019.
https://doi.org/10.18653/v1/P17

Dataset/Language		Tokens (Dev)
DE_A	German (Anselm)	45,996
DE_R	German (RIDGES)	9,712
EN	English	16,334
ES	Spanish	11,650
HU	Hungarian	16,707
IS	Icelandic	6,109
PT	Portuguese	26,749
SL_B	Slovene (Bohorič)	5,841
SL_G	Slovene (Gaj)	20,878
SV	Swedish	2,245

Table 1: Historical datasets used in our experiments and the size of their development sets. (Size of the training sets is fixed in all our experiments.)

Datasets We consider ten datasets spanning eight languages, taken from Bollmann (2019).[2] Table 1 gives an overview of the languages and the size of the development set, which we use for evaluation.

2 Model architecture

We use a standard attentional encoder–decoder architecture (Bahdanau et al., 2014) with words as input sequences and characters as input symbols.[3] Following the majority of previous work on this topic (cf. Sec. 5), we limit ourselves to word-by-word normalization, ignoring problems of contextual ambiguity. Our model consists of the following parts (which we will also refer to using the bolded letters):

- **S**ource embedding layer: transforms input characters into dense vectors.
- **E**ncoder: a single bidirectional LSTM that encodes the embedded input sequence.
- **A**ttention layer: calculates attention from the encoded inputs and the current decoder state using a multi-layer perceptron (as in Bahdanau et al., 2014).
- **T**arget embedding layer: transforms output characters into dense vectors.
- **D**ecoder: a single LSTM that decodes the encoded sequence one character at a time, using the attention vector and the embedded previous output characters as input.
- **P**rediction layer: a final feed-forward layer that linearly transforms the decoder output and performs a softmax to predict a distribution over all possible output characters.

Hyperparameters We tuned our hyperparameters on the English development section. We use randomly initialized embeddings of dimensionality 60, hidden layers of dimensionality 300, a dropout of 0.2 and a batch size of 30. We train the model for an unspecified number of epochs, instead relying on early stopping on a held-out validation set. Since we experiment with varying amounts of training data, we choose to derive this held-out data from the given training set, using only 90% of the tokens as actual training data and the remaining 10% to determine early stopping.

3 Multi-task learning

Multi-task learning (MTL) is a technique to improve generalization by training a model jointly on a set of related tasks. We follow the common approach of hard parameter sharing suggested by Caruana (1993), in which certain parts of a model architecture are shared across all tasks, while others are kept distinct for each one. Such approaches have been applied successfully to a variety of problems, e.g., machine translation (Dong et al., 2015), sequence labelling (Yang et al., 2016; Peng and Dredze, 2017), or discourse parsing (Braud et al., 2016).

Auxiliary tasks We experiment with the following auxiliary tasks:

- **Autoencoding.** We use data extracted from Wikipedia[4] and train our model to recreate the input words. In the normalization task, large parts of the input words often stay the same, so autoencoding might help to reinforce this behavior in the model.
- **Grapheme-to-phoneme mapping (g2p).** This task uses the data by Deri and Knight

[2] The datasets are available from: https://github.com/coastalcph/histnorm

[3] Our implementation uses the XNMT toolkit (Neubig et al., 2018, https://github.com/neulab/xnmt).

[4] Whenever possible, we used the dumps provided by the Polyglot project: https://sites.google.com/site/rmyeid/projects/polyglot
Since an Icelandic text dump was not available from Polyglot, we generated one ourselves using the Cirrus Extractor: https://github.com/attardi/wikiextractor
All dumps were cleaned from punctuation marks.

(2016) to map words (i.e., sequences of graphemes) to sequences of phonemes. Bollmann et al. (2017) previously showed that this task can improve historical normalization, possibly because changes in spelling are often motivated by phonological processes, an assumption also made by other normalization systems (Porta et al., 2013; Etxeberria et al., 2016).

- **Lemmatization.** We use the UniMorph dataset (Kirov et al., 2018)[5] to learn mappings from inflected word forms to their lemmas. This task is similar to normalization in that it maps a set of different word forms to a single target form, which typically bears a high resemblance to the input words.

Since we train separate models for each historical dataset, we always use auxiliary data from the same language as the dataset.

Training details When training an MTL model, we make sure that each training update is based on a balanced combination of main and auxiliary task inputs; i.e., for each batch of 30 tokens of the historical normalization task, the model will see 10 tokens from each auxiliary task. Epochs are still counted based on the normalization task only. This way, we try to make up for the imbalanced quantity of different auxiliary datasets.

3.1 Experiment 1: What to share?

In previous work on multi-task learning, there is no clear consensus on which parts of a model to share and which to keep separate. Bollmann et al. (2017) share all parts of the model except for the final prediction layer, while other multi-task sequence-to-sequence models keep task-specific encoders and decoders (cf. also Sec. 5). In principle, though, the decision to share parameters between tasks can be made for each of the encoder–decoder components individually, allowing for many more possible MTL configurations.

Setup We explore the effect of different sharing configurations. The architecture described in Sec. 2 leaves us with $2^6 = 64$ possible model configurations. When all parameters are shared, this is identical to training a single model to perform all tasks at once; when none are shared, this is identical to a single-task model trained on historical normalization only. We identify an MTL configuration using letters (cf. the bold letters from Sec. 2) to indicate which parts of the model are shared; e.g., an "SE" model would share the source embeddings and the encoder, an "SEATD" model would share everything except the final prediction layer, and so on.

In Experiment 1, we only use the first 1,000 tokens of the English historical dataset for training. We combine this with all three auxiliary tasks (using their full datasets) and train one MTL model for each of the 64 different sharing configurations.

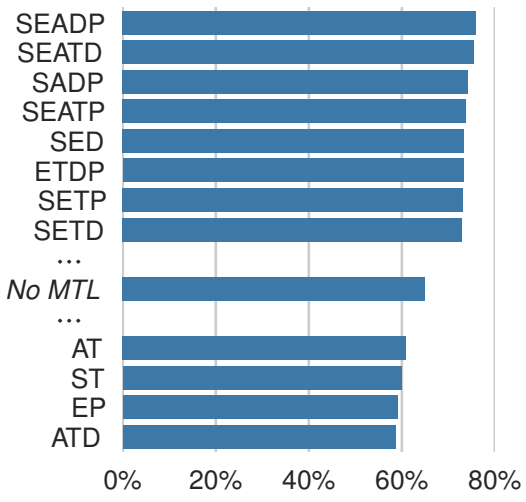


Figure 1: Normalization accuracy on the English-1k dataset, trained jointly with all three auxiliary tasks; letters indicate which model components (cf. Sec. 2) are shared between tasks.

Results Figure 1 shows an excerpt of the results, evaluated on the dev set of the English dataset. The best MTL model achieves a normalization accuracy of 75.9%, while the worst model gets 58.6%. In total, 49 configurations outperform the single-task model, showing the general effectiveness of the MTL approach. Sharing more is generally better; nine out of the top ten configurations share at least four components. Figure 2 visualizes the accuracy distribution by the number of shared components in the MTL model, supporting this conclusion.

3.2 Experiment 2: Which auxiliary tasks?

In the previous experiment, we trained the models using all three auxiliary tasks at the same time. However, not all of these tasks might be equally helpful for learning the normalization task. While Bollmann et al. (2017) show the effectiveness of

[5] https://unimorph.github.io/

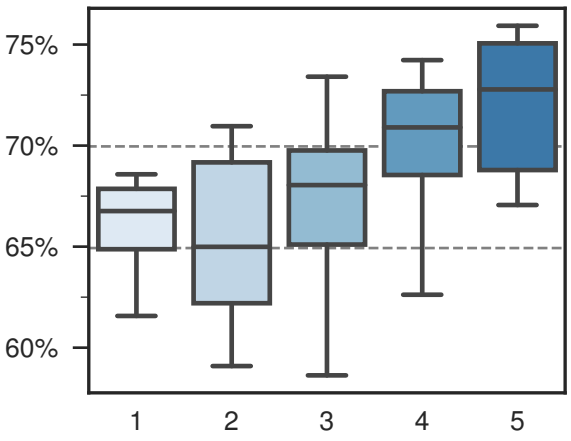


Figure 2: Quartiles of the normalization accuracies (on English-1k) by the number of shared components in the MTL model; bottom dashed line indicates no shared components (= single-task), top dashed line indicates all (= 6) shared components.

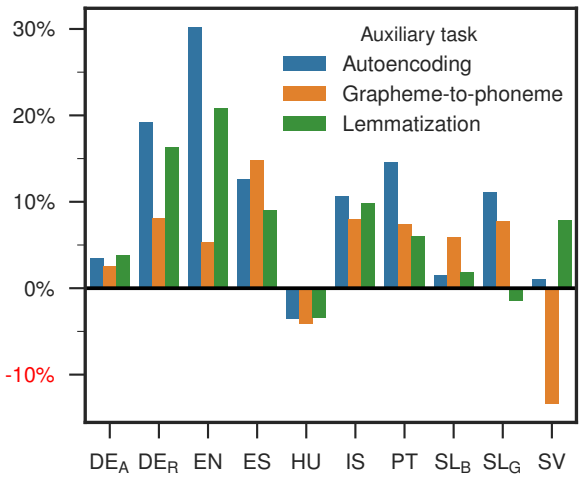


Figure 3: Error reduction for the SEADP configuration by auxiliary task, using 1,000 tokens from the historical datasets for training.

the grapheme-to-phoneme task, they only evaluate on German, and autoencoding and lemmatization have so far not been evaluated at all for improving historical text normalization.

Setup We want to investigate the improvements from each auxiliary task in isolation compared to (a) the single-task baseline and (b) the previous approach of training with all three auxiliary tasks simultaneously. For this, we select the best MTL configuration from Sec. 3.1, which is to share everything except the target embeddings ("SEADP"), and train one single-task model and four MTL models per dataset: one for each of the three auxiliary tasks, and one that uses all three tasks at the same time.

As before, we only use the first 1,000 tokens of each historical dataset. This also makes the results more comparable across datasets, as the size of the training set for the main task can affect the usefulness of multi-task learning.[6]

Results Figure 3 shows the error reduction of the MTL models compared to the single-task setup. For most datasets, MTL improves the results; the main exception is Hungarian, where all three auxiliary tasks lead to a loss in accuracy. The results show that not all auxiliary tasks are equally beneficial. Autoencoding provides the largest error reduction in most cases, while lemmatization is often slightly worse, but provides the best result for German (Anselm) and Swedish. The grapheme-to-phoneme task, on the other hand, performs worst on average, yielding much less benefits on German (Ridges) and English, and even *increases* the error on Swedish.

Table 2a shows the accuracy scores for all datasets and models. The full MTL model—training jointly on all tasks—only achieves the best performance on four of the datasets. Since the dev sets used for this evaluation vary strongly in size, we also calculate the *micro-average* of the accuracy scores, i.e., the accuracy obtained over the concatenation of all datasets. Here, we can see that using only autoencoding as an auxiliary task actually produces the highest average accuracy.

3.3 Experiment 3: How much training data?

All previous experiments have used 1,000 tokens from each historical dataset for training. Bollmann et al. (2018) show that the benefits of multi-task learning depend on training data quantity, so it is unclear whether the findings generalize to smaller or larger datasets.

Setup We analyze the benefit of MTL depending on the amount of training data that is used for the main task. We do this by training MTL models (using all three auxiliary tasks, as in Sec. 3.1) with varying amounts of historical training data, ranging from 100 tokens to 50,000 tokens. Different

[6]The same is true, of course, for the size of the auxiliary datasets. We try to balance out this factor by balancing the training updates as described in Sec. 3 "Training details", but we also note that we do not observe a correlation between auxiliary dataset size and its effectiveness for MTL in Fig. 3.

Dataset	Single	Multi-task			
		Autoenc	Lemma	g2p	ALL 3
DE_A	54.84	56.41	**56.55**	55.99	56.52
DE_R	56.72	**65.05**	63.79	60.25	64.49
EN	66.95	**76.94**	73.84	68.72	72.01
ES	74.68	77.87	76.97	78.45	**79.09**
HU	**42.44**	40.39	40.49	40.07	38.64
IS	63.40	67.31	67.02	66.31	**68.51**
PT	72.23	**76.28**	73.89	74.27	75.55
SL_B	74.06	74.44	74.54	**75.59**	74.39
SL_G	86.34	87.86	86.15	87.40	**89.45**
SV	69.98	70.29	72.34	65.97	**73.05**
Micro-Avg	64.46	**67.46**	66.47	65.79	67.04

(a) Single-task vs. multi-task models

Dataset	Best in (a)	from Bollmann (2019)		
		Norma	SMT	NMT
DE_A	56.55	**61.27**	58.60	52.74
DE_R	65.05	73.62	**75.04**	60.61
EN	76.94	**84.53**	83.81	66.93
ES	79.09	**86.21**	85.89	76.32
HU	42.44	**55.75**	53.00	40.52
IS	68.51	70.86	**72.30**	62.80
PT	76.28	**82.94**	82.00	71.43
SL_B	75.59	78.97	**82.90**	73.83
SL_G	89.45	84.36	**90.00**	86.31
SV	73.05	74.54	**78.51**	66.43
Micro-Avg	68.13	**73.30**	73.07	63.80

(b) Comparison to previous work

Table 2: Normalization accuracy on dev sets after training on 1,000 tokens. Best results highlighted in bold.

sharing configurations might conceivably give different benefits based on the training set size. We therefore evaluate each of the top three MTL configurations from Sec. 3.1, as well as the single-task model, across different data sizes.

Results Figure 4 shows learning curves for all of our historical datasets. The quantity of improvements from MTL differs between datasets, but there is a clear tendency for MTL to become less beneficial as the size of the normalization training set increases. In some cases, using MTL with larger training set sizes even results in *lower* accuracy compared to training a single-task model to do normalization only. This suggests that multi-task learning—at least with the auxiliary tasks we have chosen here—is mostly useful when the training data for the main task is sparse.

Since the accuracy scores of the different models are often within close range of each other, Figure 5 visualizes the three MTL configurations in terms of error reduction compared to the single-task model, averaged over all ten datasets. This again highlights the decreasing gains from MTL with increasing amounts of training data.

3.4 Comparison to previous work

Bollmann (2019) compares normalization models when trained with different amounts of data, including a setting with 1,000 tokens for training, allowing us to directly compare our results with those reported there.[7] These results are shown in Table 2b. Comparing our single-task system with their NMT model (which is very similar to ours), we see that the scores are overall comparable, suggesting that our implementation is sound. At the same time, our best scores with MTL are still far below those produced by SMT or the rule-based "Norma" tool. This, unfortunately, is a negative result for the neural approach in this low-resource scenario, and the diminishing gains from MTL that were shown in Sec. 3.3 suggest that our presented approach will not be sufficient for elevating the neural model above its non-neural alternatives for this particular task.

3.5 Experiment 4: Zero-shot learning

Most previous work on historical text normalization has focused on a supervised scenario where some labeled data is available for the target domain, i.e., the particular historical language you are interested in. Since spelling variation is highly idiosyncratic in the absence of normative spelling guidelines, models are not expected to generalize beyond specific language stages, or sometimes even manuscript collections. This means that many historical text normalization projects require resources to annotate new data. This paper is the first to experiment with a zero-shot learning scenario that leverages existing data from other languages, but assumes *no* labeled data for the target language.

Setup For the zero-shot experiments, we use the same model as for the single-task baseline; in other words, all layers are shared between all tasks and languages. Instead, to allow the model to discern between languages and tasks, we prepend two extra symbols to all model inputs: a *lan-*

[7] Bollmann (2019) only shows graphical plots for these results, but the exact figures were released at: https://github.com/coastalcph/histnorm/blob/master/appendix_tab6.pdf

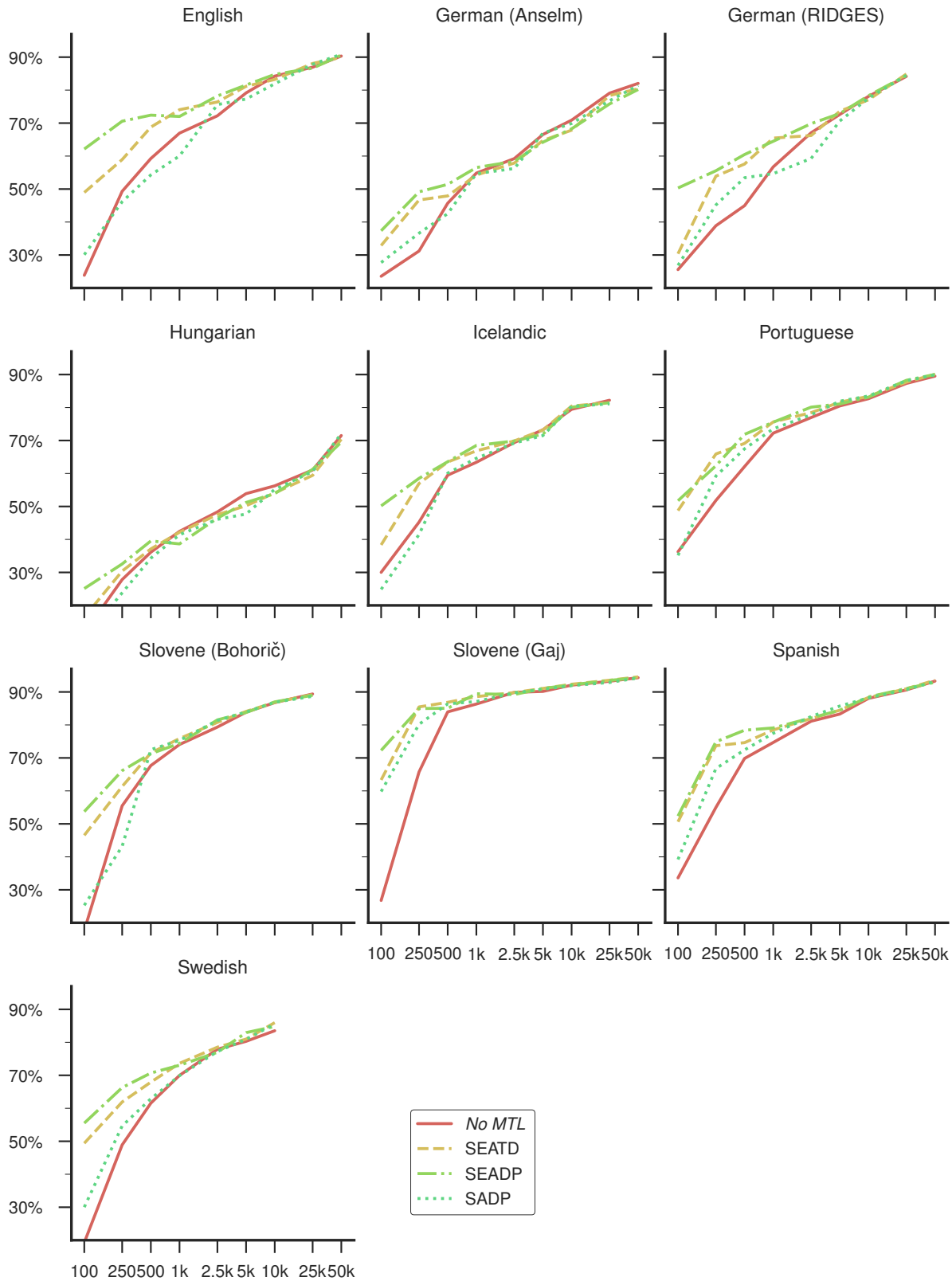


Figure 4: Learning curves for all datasets, showing the normalization accuracy of a single-task and three multi-task learning models in relation to the training set size; note that the x-axis is log-scaled.

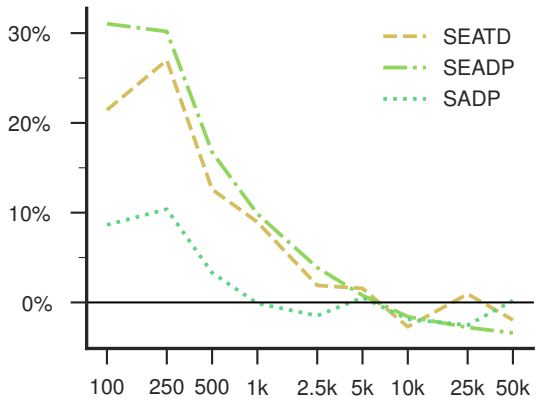


Figure 5: Error reduction for three MTL configurations by training set size, (micro-)averaged over all datasets.

guage identifier and a *task identifier.* For each language, we then train a single model on all tasks—normalization, lemmatization, autoencoding, and grapheme-to-phoneme transduction—and all languages, *except* for the normalization task of the target language. This way, the model can observe data from the normalization task (albeit in other languages) and from the target language (albeit from auxiliary tasks only), but does not see any normalization data from the target language. In those cases where there are two datasets from the same language, we leave out *both* of them from the training step. The model is similar to previous work on zero-shot neural machine translation (Johnson et al., 2016).

As before, we include only 1,000 tokens from each historical dataset for training. In each training update, we use an equal number of samples from each dataset/task combination, and define an epoch to consist of 1,000 samples from each of these combinations. Since we do not want to feed the model any normalization data from the target language during training, we cannot use early stopping, but instead train for a fixed number of 10 epochs.

Results Table 3 shows the accuracy of zero-shot normalization compared to the naive *identity baseline,* i.e., the accuracy obtained by simply leaving the input word forms unchanged. The zero-shot approach improves over this baseline for half of the datasets, sometimes by up to 12 percentage points (DE_R). Micro-averaging the results shows an overall advantage for zero-shot learning.

Dataset	Identity	Zero-shot
DE_A	30.16	**40.94**
DE_R	43.57	**55.92**
EN	**75.47**	56.31
ES	**72.29**	64.39
HU	17.81	**20.58**
IS	**47.77**	42.95
PT	65.18	**67.64**
SL_B	39.84	**50.21**
SL_G	**85.58**	84.99
SV	**59.24**	50.65
Micro-Avg	50.17	**52.96**

Table 3: Normalization accuracy on dev sets for zero-shot experiments. Best results highlighted in bold.

4 Analysis

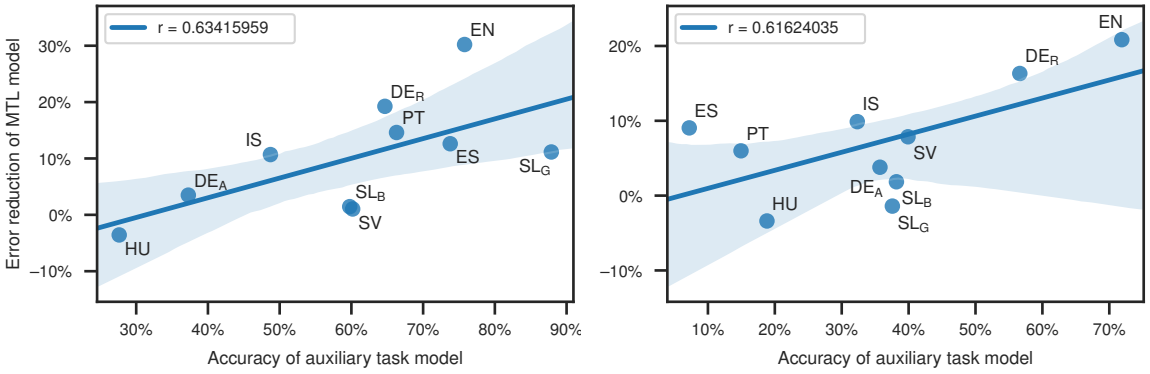
The experiment in Sec. 3.2 has shown that not all auxiliary tasks are equally useful; furthermore, autoencoding is, on average, the most useful auxiliary task of the three, closely followed by lemmatization. This gives rise to the hypothesis that MTL mostly helps the model learn the identity mappings between characters.

To analyze this, we feed the historical data into the *auxiliary models;* i.e., we treat them *as if* they were a historical text normalization model. We then correlate their normalization accuracy with the error reduction over the baseline of the MTL model using this auxiliary task. Figure 6a shows a strong correlation for the autoencoding task, suggesting that the synergy between autoencoding and historical text normalization is higher *when the two tasks are very related.* Figure 6b shows the same correlation for lemmatization.

We can also compare the error reduction from MTL to the identity baseline (cf. Tab. 3). Figure 7 shows the correlation of these scores for the full MTL model trained with all three auxiliary tasks.[8] The strong correlation suggests that the regularization effect introduced by MTL is particularly helpful with tasks where there is a strong similarity between input and output; or, in other words, that *multi-task learning prevents the model from over-generalizing* based on the training data.

The previous correlation scores only consider the performance of models trained on 1,000 tokens of historical data. Sec. 3.3 showed that the benefit of MTL diminishes when the size of the historical training sets gets larger. Figure 8 presents learning

[8]The correlation is similar when using longest common subsequence or Levenshtein distance instead of accuracy.



(a) Autoencoding

(b) Lemmatization

Figure 6: Correlations (with 95% confidence intervals) between the performance of an auxiliary task model applied to normalization data and the error reduction when using this task in a multi-task learning setup.

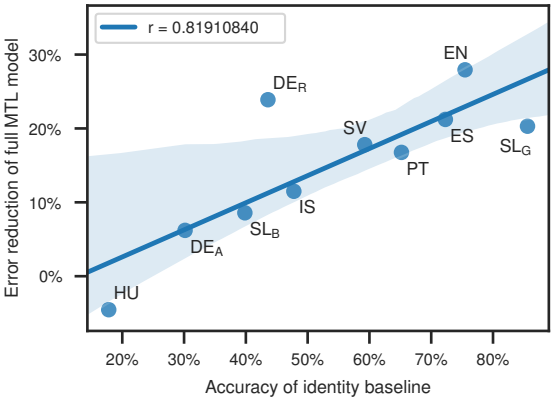


Figure 7: Correlation (with 95% confidence interval) between the identity baseline and the error reduction of the full MTL model with all three auxiliary tasks.

curves that have been micro-averaged over all ten datasets, but evaluated on different subsets of the data: (a) tokens that have been seen during training ("knowns") or not ("unknowns"); and (b) tokens that stay identical in the reference normalization or not. On average, the performance of the MTL models is comparable to that of the single-task model for known tokens and non-identity normalizations. In other words, most of the gain from MTL comes from helping the model learn the identity mappings, which becomes less relevant the more historical training data is available.

5 Related work

On previous approaches to historical text normalization, Bollmann (2019, Sec. 2) gives an extensive overview. Common approaches include rule-based algorithms—with either manually crafted or automatically learned rules—or distance metrics to compare historical spellings to modern lexicon forms (Baron and Rayson, 2008; Bollmann, 2012; Pettersson et al., 2013a). Finite-state transducers are sometimes used to model this, but also to explicitly encode phonological transformations which often underlie the spelling variation (Porta et al., 2013; Etxeberria et al., 2016).

Character-based statistical machine translation (CSMT) has been successfully applied to normalization on many languages (Pettersson et al., 2013b; Scherrer and Erjavec, 2016; Domingo and Casacuberta, 2018); neural encoder–decoder models with character-level input can be seen as the neural equivalent to the statistical MT approach (Bollmann et al., 2017; Tang et al., 2018) and have been shown to be competitive with it (Robertson and Goldwater, 2018; Hämäläinen et al., 2018), although Bollmann (2019) suggests that they are still inferior to CSMT in low-resource scenarios.

All these methods rely on individual word forms as their input; there is almost no work on incorporating sentence-level context for this task (but cf. Jurish, 2010).

MTL architectures In Sec. 3.1, we explored *what to share* between tasks in our multi-task architecture. A common approach is to share only the first layers (e.g., Yang et al., 2016; Peng and Dredze, 2017). Multi-task encoder–decoder models will often keep the whole encoder and decoder task- or language-specific (Dong et al., 2015; Luong et al., 2015). Firat et al. (2016) explore the effect of sharing the attentional component across

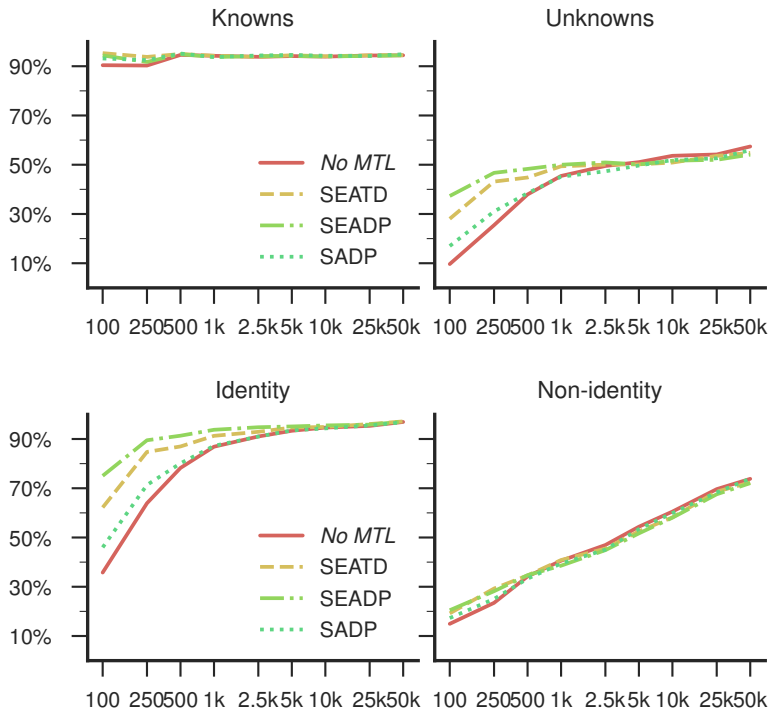


Figure 8: Learning curves, micro-averaged over all datasets, for different subsets of the data.

all languages, while Anastasopoulos and Chiang (2018) compare both parallel and cascading model configurations.

A different MTL approach is to share *all* parts of a model, but prepend a task-specific symbol to the input string to enable it to learn task-specific features (cf. Sec. 3.5). Milde et al. (2017) use this approach for grapheme-to-phoneme conversion; Kann et al. (2017) apply it to morphological paradigm completion.

Auxiliary tasks for MTL For *which auxiliary task(s) to use* (Sec 3.2), few systematic studies exist. Most approaches use tasks that are deemed to be related to the main task—e.g., combining machine translation with syntactic parsing (Kiperwasser and Ballesteros, 2018)—and justify their choice by the effectiveness of the resulting model. Bingel and Søgaard (2017) analyze beneficial task relations for MTL in more detail, but only consider sequence labelling tasks. For zero-shot learning (Sec. 3.5), we use an architecture very similar to Johnson et al. (2016), also used for grapheme-to-phoneme mapping in Peters et al. (2017).

6 Conclusion

We performed an extensive evaluation of a neural encoder–decoder model on historical text normalization, using little or even no training data for the target language, and using multi-task learning (MTL) strategies to improve accuracy. We found that sharing more components between main and auxiliary tasks is usually better, and autoencoding generally provides the most benefit for our task. Analysis showed that this is mainly because MTL helps the model learn that most characters should stay the same, and that its beneficial effect vanishes as the size of the training set increases. While our models did not beat the non-neural models of Bollmann (2019), we believe our work still provides interesting insights into the impact of MTL for low-resource scenarios.

Acknowledgments

We would like to thank the anonymous reviewers of this as well as previous iterations of this paper for several helpful comments.

This research has received funding from the European Research Council under the ERC Starting Grant LOWLANDS No. 313695. Marcel Bollmann was partly funded from the European Union's Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 845995.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 82–91, New Orleans, Louisiana. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Alistair Baron and Paul Rayson. 2008. VARD 2: A tool for dealing with spelling variation in historical corpora. In *Proceedings of the Postgraduate Conference in Corpus Linguistics*.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169, Valencia, Spain. Association for Computational Linguistics.

Marcel Bollmann. 2012. (Semi-)automatic normalization of historical texts using distance measures and the Norma tool. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2)*, Lisbon, Portugal.

Marcel Bollmann. 2018. Normalization of historical texts with neural network models. *Bochumer Linguistische Arbeitsberichte*, 22.

Marcel Bollmann. 2019. A large-scale comparison of historical text normalization systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3885–3898. Association for Computational Linguistics.

Marcel Bollmann, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 332–344, Vancouver, Canada. Association for Computational Linguistics.

Marcel Bollmann, Anders Søgaard, and Joachim Bingel. 2018. Multi-task learning for historical text normalization: Size matters. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 19–24. Association for Computational Linguistics.

Chloé Braud, Barbara Plank, and Anders Søgaard. 2016. Multi-view and multi-task training of RST discourse parsers. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1903–1913, Osaka, Japan. The COLING 2016 Organizing Committee.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the 10th International Conference on Machine Learning (ICML)*, pages 41–48.

Aliya Deri and Kevin Knight. 2016. Grapheme-to-phoneme models for (almost) any language. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 399–408, Berlin, Germany. Association for Computational Linguistics.

Miguel Domingo and Francisco Casacuberta. 2018. Spelling normalization of historical documents by using a machine translation approach. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, pages 129–137.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China. Association for Computational Linguistics.

Izaskun Etxeberria, Iñaki Alegria, Larraitz Uria, and Mans Hulden. 2016. Evaluating the noisy channel model for the normalization of historical texts: Basque, Spanish and Slovene. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1064–1069, Portorož, Slovenia. European Language Resources Association (ELRA).

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *CoRR*, abs/1601.01073.

Mika Hämäläinen, Tanja Säily, Jack Rueter, Jörg Tiedemann, and Eetu Mäkelä. 2018. Normalizing early english letters to present-day english spelling. In *Proceedings of the Second Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 87–96. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558.

Bryan Jurish. 2010. More than words: using token context to improve canonicalization of historical German. *Journal for Language Technology and Computational Linguistics*, 25(1):23–39.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. One-shot neural cross-lingual transfer for paradigm completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1993–2003, Vancouver, Canada. Association for Computational Linguistics.

Akisato Kimura, Zoubin Ghahramani, Koh Takeuchi, Tomoharu Iwata, and Naonori Ueda. 2018. Few-shot learning of neural networks from scratch by pseudo example optimization. *arXiv e-prints*, abs/1802.03039.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Graldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Arya D. McCarthy, Sandra Kbler, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. UniMorph 2.0: Universal morphology. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 1868–1873. European Language Resources Association (ELRA).

Natalia Korchagina. 2017. Normalizing medieval German texts: from rules to deep learning. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*, pages 12–17, Gothenburg. Linköping University Electronic Press.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *CoRR*, abs/1511.06114.

Benjamin Milde, Christoph Schmidt, and Joachim Khler. 2017. Multitask sequence-to-sequence models for grapheme-to-phoneme conversion. In *Proceedings of Interspeech 2017*, pages 2536–2540.

Graham Neubig, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, Sarguna Padmanabhan, Ye Qi, Devendra Singh Sachan, Philip Arthur, Pierre Godard, John Hewitt, Rachid Riad, and Liming Wang. 2018. XNMT: The extensible neural machine translation toolkit. In *Conference of the Association for Machine Translation in the Americas (AMTA) Open Source Software Showcase*, Boston.

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 91–100, Vancouver, Canada. Association for Computational Linguistics.

Ben Peters, Jon Dehdari, and Josef van Genabith. 2017. Massively multilingual neural grapheme-to-phoneme conversion. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 19–26, Copenhagen, Denmark. Association for Computational Linguistics.

Eva Pettersson, Beáta Megyesi, and Joakim Nivre. 2013a. Normalisation of historical text using context-sensitive weighted Levenshtein distance and compound splitting. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, pages 163–179, Oslo, Norway. Linköping University Electronic Press.

Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. 2013b. An SMT approach to automatic annotation of historical text. In *Proceedings of the Workshop on Computational Historical Linguistics at NODALIDA 2013*, NEALT Proceedings Series 18, pages 54–69. Linköping University Electronic Press.

Michael Piotrowski. 2012. *Natural Language Processing for Historical Texts*. Number 17 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool.

Jordi Porta, José-Luis Sancho, and Javier Gómez. 2013. Edit transducers for spelling variation in Old Spanish. In *Proceedings of the Workshop on Computational Historical Linguistics at NODALIDA 2013*, NEALT Proceedings Series 18, pages 70–79. Linköping University Electronic Press.

Alexander Robertson and Sharon Goldwater. 2018. Evaluating historical text normalization systems: How well do they generalize? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 720–725. Association for Computational Linguistics.

Yves Scherrer and Tomaž Erjavec. 2016. Modernising historical Slovene words. *Natural Language Engineering*, 22(6):881–905.

Gongbo Tang, Fabienne Cap, Eva Pettersson, and Joakim Nivre. 2018. An evaluation of neural machine translation models on historical spelling normalization. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1320–1331, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *CoRR*, abs/1603.06270.